3D Mapping for a Reliable Long-Term Navigation

Jonathan Ginés[1], Francisco Martín[1(✉)], Vicente Matellán[2], Francisco J. Lera[3], and Jesús Balsa[2]

[1] University of Rey Juan Carlos, Móstoles, Spain
j.gines@alumnos.urjc.es, francisco.rico@urjc.es
[2] University of León, León, Spain
{vicente.matellan,jesus.balsa}@unileon.es
[3] Computer Science and Communications Research Unit (CSC), University of Luxembourg, Luxembourg city, Luxembourg
francisco.lera@uni.lu

Abstract. The use of maps allows mobile robots to navigate between known points in an environment. Using maps allows to calculate routes avoiding obstacles and not being stuck in dead ends. This paper shows how to integrate 3D perceptions on a map to obtain obstacle-free paths when obstacles are not at the level of 2D sensors, but elevated. Chairs and tables usually pose a problem when one can only see the legs with a 2D laser, although they present a high hurdle with a much larger area. This approach builds a static map starting from the construction plans of a building. A long-term map is started from the static map, and updated when adding and removing furniture, or when doors are opened or closed. A short-term map represents dynamic obstacles such as people. Obstacles are perceived by merging all available information, both 2D laser and RGB-D cameras, into a compact 3D probabilistic representation. This approach is appropriate for fast deployment and long-term operations in office or domestic environments, able to adapt to changes in the environment. This work is designed for domestic environments, and has been tested in the RoboCup@home competition, where robots must navigate in an environment that changes during the tests.

Keywords: Long-term navigation · 3D mapping · Mobile robot · Robocup

1 Introduction

Mobile robots navigate along the environment to perform the commanded tasks. This capability is critic for the success of many applications, and it has to be accomplished robustly. One of the elements of navigation is the map. Maps represent the structure of the environment, the obstacles and the free space. Using this information, robots update their pose and generate paths. Usually,

A. Ollero et al. (eds.), *ROBOT 2017: Third Iberian Robotics Conference*, Advances in Intelligent Systems and Computing 694, https://doi.org/10.1007/978-3-319-70836-2_24

maps remain unchanged once generated. This is not convenient in domestic environment when operating for days because furniture can change its position and new objects can appear. Even for short-term operation, doors can be open or closed.

For this reason, we present a mapping method for long time operation in domestic environment. Simultaneous Localization and Mapping (SLAM) techniques also updates the map while operation, but they are more focused in building maps, instead of deploying a robot in a known environment and start operating immediately. Besides of this, SLAM techniques require a robustness hard to get. Walls, doors and corridors are known a priori. Usually, we can obtain an architectonic map of large environments likes offices, universities, public buildings or hotels.

Our proposal is a mapping method starts from these kind of structural information of walls, doors and corridors, building a static map which never changes. This information is enough to maintain a good estimation about the robot pose, when starting pose is known. Obstacles not present in the static part are incorporated to a short-term map. When this obstacle is persistent in time, it is incorporated to a long-term map. In the same way, if a obstacle disappear, it is removed from the long-term map. The robot uses the combined map of static and long-term information to self-localize and build routes.

Mobile robots that travel on the ground often use 2D maps, since the z component, or the *roll* or *pitch* are not taken into account to locate or generate routes. For most navigation tasks, using a 2D laser is considered appropriate. We have found that it is not enough, because a robot is a 3D volume that must pass through an environment that may present obstacles that do not necessarily have to be at the height of the laser. Using a 2D approach, we had problems with shelves, tables, chairs and all kinds of obstacles that are partially detected at the height of the laser. For this reason, one of the main contributions of our work is to merge the information of a 2D laser with any other sensor, mainly 3D cameras, currently used in many robots to have a complete 3D information which let us to safety navigate.

This method is applied in the RoboCup@home [1], RoCKIn@home [2] and European Robotics League (ERL) competitions. These competitions propose several tasks in a domestic environment which simulates a house. The technologies evaluated in the test includes navigation, manipulation, object/people recognition, communication with humans and task planning. These testbeds let us to compare our approach with the other participants' solutions in the same conditions.

This paper is organized as follows: in the Sect. 2 we describe the relevant work in the area, with special attention to the last method used in the competition. The description of our contribution is in Sect. 3, where we briefly describe the ROS navigation system and present our mapping system. The experimental validation is in Sect. 4, where the experiments in the simulator confirm the result in the competition. Finally, we discuss the results and work in Sect. 5.

2 Related Work

Mapping in the form of a grid has been a widely used approach. Most mobile robots move on the ground, so they do not need more info to self-locate and navigate than a 2D grid of obstacles. It is a compact and general way of representing the environment. Maybe one of the first successful works using this approach is [3], and a full description of capabilities is shown in [4]. It is a widely used mapping method in conjunction with self localization algorithms such as Monte Carlo [5,6] or Markov [7], when sensory information can be reduced to 2D readings of obstacles. Most of these approaches start from a premise that is not fulfilled when the operation of the robot can last for days or months, as the environment can change. These jobs usually do the mapping once, and assume that the environment never changes. In fact, if the robot is in an crowded environment, techniques such as using the ceiling image as a map are used [8].

In [9], robots operation last for 1000 Km [10]. This approach takes as its basis an architectural map using a Bayesian network to decide what type of feature is being detected. In our case, we simply model the features on different maps. Our approach is simpler, but equally effective. By not having to detect different types of obstacles, our method is more general and scalable.

Long-term navigation should keep in mind that environments may vary. In our opinion, the only thing that does not change is the structure of the building, meaning the walls The rest can change, and the robot has to adapt to calculate better routes. In [11] a Dynamic Pose Graph SLAM is used to map low dynamic environments. In this work, 2D maps are generated from a graph storing changes in the environment. The authors state that localization is improved using this type of maps. In [12] a work is presented in which are maintained spatial-temporal 3D maps. Each position stores the probability that a node is busy depending on the time of day. In this work a robot plans which points to visit depending on the entropy of surrounding areas, since this occupation is repeated cyclically. Our work is focused on providing a map which allows us to generate good routes to navigate. Adding these temporal characteristics is future work and not included in the presented approach.

The appearance of 3D sensors has given rise new types of 3D maps [13]. Octomap [14] is a compact way to encode the environment about obstacles and even colors. These maps are very useful for self-localization [15], although not so much to navigation when the robot is on the ground and there are no obstacles in height that prevent the robot from passing through.

3 Mapping System

Our mapping system is integrated in the navigation stack of ROS. The standard ROS navigation system is designed for static environments. If a little change is produced in the scenario, the localization component bear with this but when significant portions of the environment change, it can not deal with this situation. This system has also problems dealing with doors, that sometimes can be open

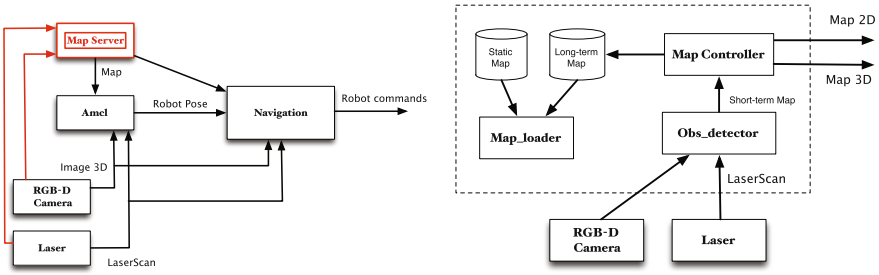

Fig. 1. ROS navigation system with our contributions in red (left). Diagram of the proposed dynamic map module (right).

and sometimes closed, objects smaller than the position of the laser sensors and tables because the robot can only perceive the thin legs of it.

To deal with these problems, we have replaced the original `map_server` by a node that implements our approach, as is shown in the left side of the Fig. 1.

The module `map_server` is a critical component of the navigation stack of ROS. It provides of occupancy information both to the navigation module, which actually makes plans and send commands to the actuators, and the Amcl module, which localizes the robot using a Monte Carlo [16] algorithm called KLD-Sampling [17].

For representing occupancy maps in 3D, we use a compact representation called *octomap* [14], based on octrees (Fig. 2).

Internally, our map module uses this 3D representation of the environment. As we use the existing navigation and localization modules in the ROS navigation stack, and their inputs are 2D occupation maps, our module produces two outputs: the final 3D occupancy map and a 2D occupancy map, built by flattening the 3D map. This flattening process gets the value of each (x, y) cell as the maximum occupancy value of cells $(x, y, z) \forall z \in [z_{min}, z_{max}]$.

We want our robot to navigate days, weeks, or months through an environment without remapping. We believe that the only really static part of an environment is its walls. Anything else can change its position over time. For this reason our robot starts from metric measures or from the architectonics maps,

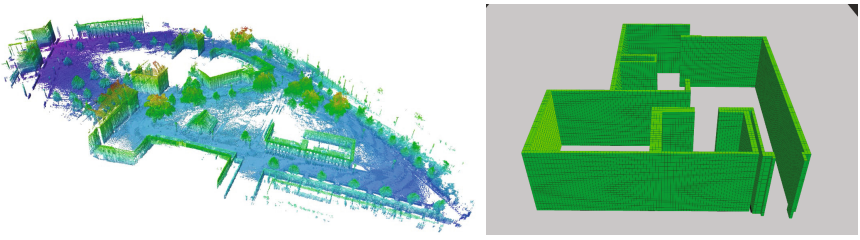

Fig. 2. 3D occupancy representation using octomaps.

which we will call a *static map*. This map include the unchanged parts of the scenario. All maps are 3D maps, as we'll describe in the next point, and as a final step we converts them in a 2D occupancy grid.

When the robot perceives an obstacle that is not on the static map, it incorporates it into a *short-term map* whose 3D cells value varies rapidly over time. If it is a person, it will quickly disappears and the *short-term map* will reflect it. On the other hand, it is a new persistent obstacle, the robot will perceive it for a long time and because of this the obstacle will become part of a *long-term map*. Similarly, if an obstacle was on the map and it disappears, it will be removed from the *long-term map*.

The long-term map is periodically saved to disk as an .ot file. When the map server starts, it loads both the *static map* and the *long-term map*. As we previously introduced, the map server publishes the combination of these maps and convert it in a 2D occupancy grid map for the navigation and localization modules.

Internally, map server is composed by the following elements also illustrated on the right side of Fig. 1.:

- **Map_loader**: This node loads the static and the long-term maps from disk.
- **Sensors2octomap**: This node combines the information of the sensors and create an octomap with it.
- **Obs_detector**: The main function of this node is to detect objects using the octomap from the sensors. This information produces a 3D octomap, which is the main information of the short-term map.
- **Map_controller**: Node with the function of compose the final octomap, publish it in /octomap topic and update the long-term map with the values of the short-term map.
- **Octomap2map**: This node takes the information of the /octomap topic and creates a 2D occupancy grid with it.

Maps are 3D grids of values representing obstacles or free space. The dimension of each 3D cell in the grid is preconfigured and an usual value is $100\,mm \times 100\,mm$. It probabilistically represents obstacles using values in the range $[0, 1]$.

We use several maps at the same time for different:

- **static map** (map_S): The 2D map is built from metric measures or from the architectonics maps.
- **short-term map** (map_{st}): This octomap is initialized totally empty. When running, cell values are updated with the obstacle information from sensors.
- **long-term map** (map_{lt}): The long-term map includes the objects that persist in time which could affect when generating routes. This octomap is updated with the information from the short-term map.
- **resulting map** (map_f): This is the resulting octomap published by the map server.

– **projected map (map_p)**: Conversion of the map_f into a 2D map. To do this conversion we take all 3D cells and we associate each one with a cell in 2D map. The value of the 2D map cell will be the max value between all 3D nodes associated with it.

$$map^{2D} = Flat(map^3D) \tag{1}$$

$$map^{2D}_{i,j} = max(map_{i,j,k} \forall k \in [-\infty, +\infty]) \tag{2}$$

Maps are created and updated following these operations:

$$map_f = map_S \oplus map_{lt} \tag{3}$$

where $\oplus$ is defined as,

$$\begin{aligned} m1 \oplus m2 : max(m1_{i,j}, m2_{i,j}), \\ \forall m1_{i,j} \in m1, \forall m2_{i,j} \in m2 \end{aligned} \tag{4}$$

and

$$map_{lt} = map_{lt} \otimes map_{st} \tag{5}$$

where $\otimes$ is defined as:

$$\begin{aligned} m1 \otimes m2 : m1 = \begin{cases} m1_{i,j} & \text{if } m2_{i,j} < 1, \\ m1_{i,j} + 1 & \text{if } m2_{i,j} = 1 \end{cases} \\ \forall m1_{i,j} \in m1, \forall m2_{i,j} \in m2 \end{aligned} \tag{6}$$

where 1 value represent the probability that a cell is occupied.

After these operations we have set up a map (map_f) ready for being converted in 2D map for being used by localization and navigation modules.

The 2D map, map_p, is built following this operation:

$$\begin{aligned} m1_{i,j} = max(m2_{i,j,\forall k}), \\ \forall m1_{i,j} \in m1, \forall m2_{i,j,k} \in m2 \end{aligned} \tag{7}$$

4 Experiments

In order to validate the work presented in this paper, we have carried out experiments both in the simulator as the previous step for implementing in the real robot and testing it in incoming real competitions.

The goal of these experiments is to validate the proposed algorithm in a controlled environment. The fundamental aspect of our approach is that the map update is performed correctly and we can avoid small objects:

- Detection of dynamic objects should not affect the long-term map.
- The perception of static objects must be incorporated in a reasonably short time to the long-term map, anchoring itself with successive detections.
- The disappearance of static objects that were already on the long-term map should be eliminated quickly.
- Closed doors are assimilated to static obstacles that are incorporated into the long-term map, being eliminated quickly when the doors are opened.
- Avoid objects that could presents difficulties to perceive with the laser sensor, like a chairs or tables.
- Avoid smaller objects under the position of the laser sensor.

1. **Static objects**. In the first experiment, the robot perceives a static object (Fig. 3).

Fig. 3. Experiment with a static object in front of the robot.

At first, the object will be added to the short-term map and 17s later, when the value of the cells in short-term map reach 1, the long-term map will reflect this change.

Figure 4 represents the evolution of the short-term map and long-term map when the robot perceive a new object. If the object is removed from the scene, the short-term map will reflects this change clearing the cells of the object and when the short-term map its clean, the cells of the object in long-term map will starts to decrease their value.

2. **Impact of the mapping mechanism in the robot's motion**. If a door is closed just when the robot go to cross it or if a person block the way of the robot, local planner of the navigation will avoid the danger. This local planner only uses the information of the laser sensor, because of this we had to create a new laserscan to reflect the information of the combination between laser and RGB-D sensors.

Figure 5 shows the statistics parameters of the time to add and delete objects from long-term map and how much time is needed to influence in the global planner.

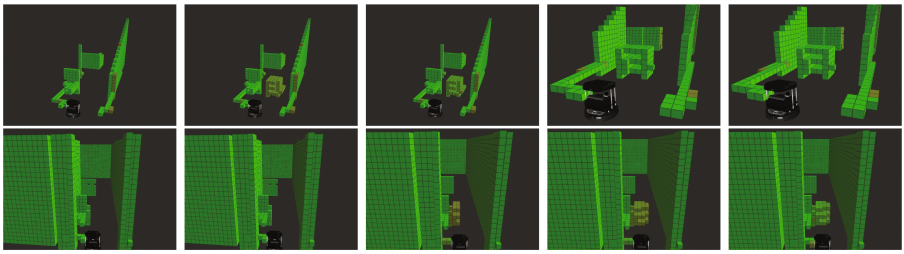

Fig. 4. Changes in cell's value in presence of static objects. Up, the short-term map and down the long-term map

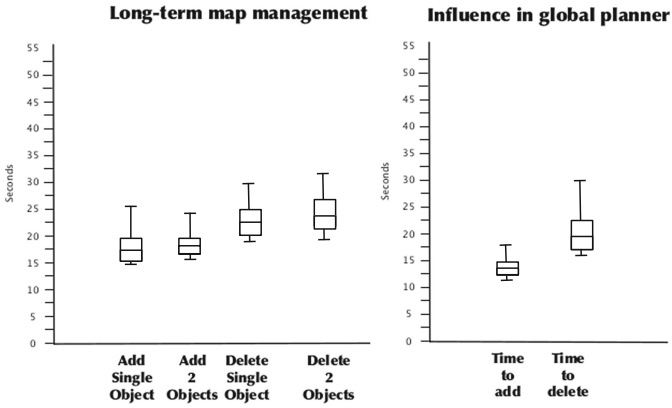

Fig. 5. Statistics parameters of the time to change the long-term map and the global planner.

In this experiment we compare the previous 2D navigation method with our approach, where RGB-D sensors are used.

We commanded the robot to go to a position beyond a table. Figure 6 shows how the robot went under the table. If the robot would taller than Kobuki, the robot would had collide with the table.

In Fig. 7 we show how the robot avoid the table using our 3D mapping system. The global planner takes the information from the 2D map, which is generated with the information of all the sensors, and reflects the object in the navigation global map. Because of this, the navigation module does not plan a new route crossing the table, like shown in Fig. 6. Now the navigation module can plan a route avoiding the obstacle.

3. **Avoiding little objects**. In a domestic environment or in the Robocup@Home you can find some objects under the laser position in the robot in your way. Our approach can avoid colliding with it while navigating.

In Fig. 8 we show how the ball is added to the octomap and how the robot can avoid the ball.

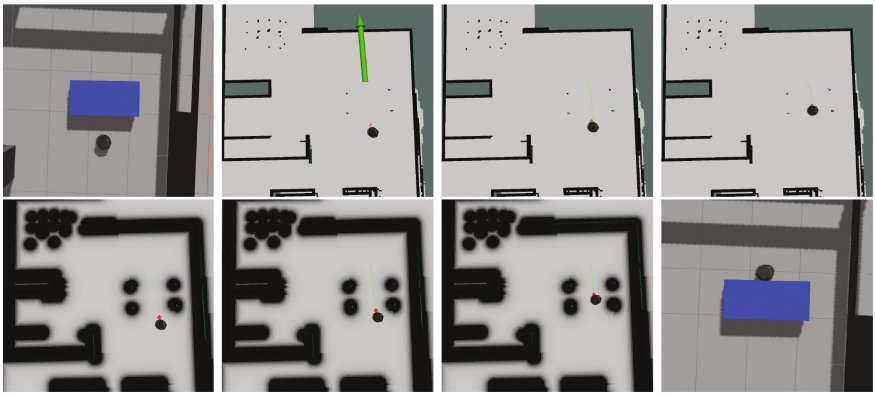

Fig. 6. Planning a new route crossing a table with 2D approach.

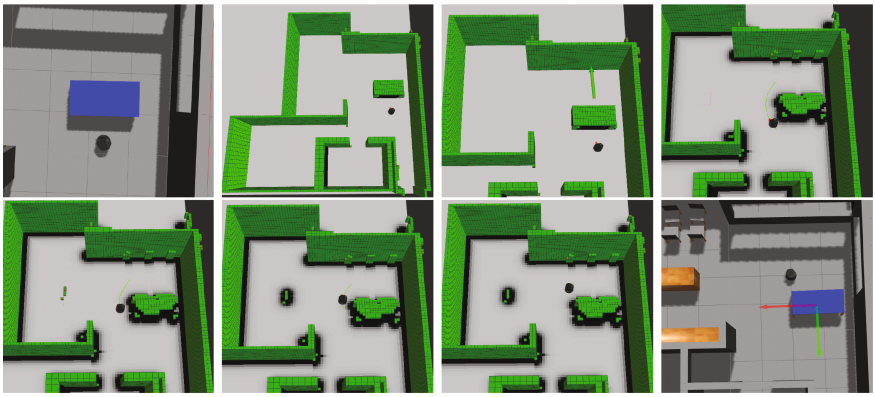

Fig. 7. Planning a new route crossing a table with 3D aproach.

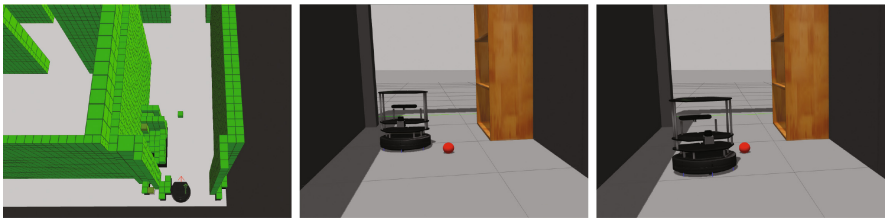

Fig. 8. Avoiding a little object.

4. **Doors opened and closed**. In this experiment, we ask the robot to go near to the blue table in the dinning room. It knows map_f ($map_S \oplus map_{lt}$), so it can compute the path: go straight to the wall, turn left and go straight again to the table. In this experiment, we have blocked this way, simulating a closed door (Fig. 9).

Fig. 9. Path blocked simulating a closed door.

The robot starts its way, but when it approximates to the closed door, map_{lt} starts to reflect this change. Path is recomputed, and the robot found an alternative way to reach its goal. The robot does not forget the closed door when the robot is looking away, because of this the robot will not use this door to calculate a new route.

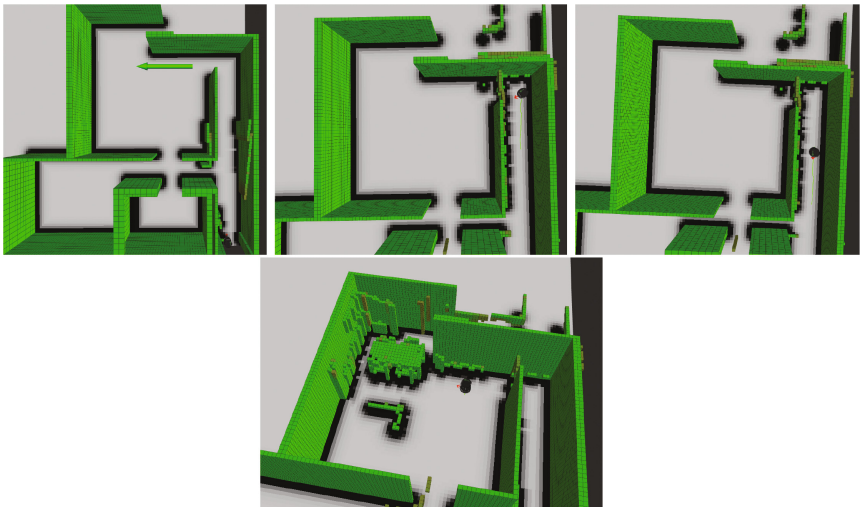

Fig. 10. Path followed.

5 Conclusions

This paper has presented our mapping system for long-term navigation, based on dynamic maps. Maps have permanent structure built from the constructions plans. All the perceived obstacles are dynamically incorporated or removed from the map, using different levels of persistency. The described approach provides a fast method for deploying a robot in a domestic environment, and for long operation in which the obstacles can modify their position.

Among the strengths of our method, we highlight, besides the fast mapping, the management of dynamic objects, the convenience to manage the opening of doors and their ability to adapt to changes in the surrounding furniture.

The main contribution of this work is the use of a 3D representation for both perceptions and the maps. This let us to perceive obstacles of any shapes, even small ones, which are not correctly perceived using only 2D laser perceptions.

The proposed method has been implemented inside the ROS navigation stack without any modification of the other modules. Because of this, the implementation of this contribution can be easily tested by the robotics community. The experiments shows the validity of this approach, and its use in competitions probes its robustness.

This work can be extended by gradually migrating the localization and navigation modules to use 3D information, although navigation benefits will only be effective in robots with 6 DoF.

Acknowledgment. This work has been supported by the Spanish Government TIN2016-76515-R Grant, supported with FEDER funds.

References

1. van der Zant, T., Wisspeintner, T.: RoboCup@Home: creating and benchmarking tomorrows service robot applications. In: Lima, P. (ed) Robotic Soccer, pp. 521–528. I-Tech Education and Publishing, Vienna (2007)
2. Lima, P.U., Nardi, D., Iocchi, L., Kraetzschmar, G., Matteucci, M.: RoCKIn@Home: benchmarking domestic robots through competitions. In: ICAR 2013, Montevideo, Uruguay (2013)
3. Thrun, S., Buckenz, A.: Integrating grid-based and topological maps for mobile robot navigation. In: Proceedings of the Thirteenth National Conference on Artificial Intelligence AAAI, Portland, Oregon, August 1996
4. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics (Intelligent Robotics and Autonomous Agents). The MIT Press, Cambridge (2005)
5. Dellaert, F., Fox, D., Burgard, W., Thrun, S.: Monte carlo localization for mobile robots. In: ICRA, vol. 2, pp. 13221328 (1999)
6. Lenser, S., Veloso, M.: Sensor resetting localization for poorly modelled mobile robots. In: International Conference on Robotics and Automation (2000). ISBN: 0780358864
7. Koenig, S., Simmons, R.: Xavier: a robot navigation architecture based on partially observable markov decision process models. In: Artificial Intelligence Based Mobile Robotics: Case Studies of Successful Robot Systems, pp. 91122 (1998)

8. Thrun, S., Bennewitz, M., et al.: MINERVA: a second-generation museum tour-guide robot. In: Proceedings of IEEE International Conference on Robotics and Automation (ICRA99) (1999)
9. Biswas, J., Veloso, M.: Episodic non-Markov localization. Robot. Auton. Syst. **87**, 162–176 (2016)
10. Biswas, J., Veloso, M.: The 1,000-km challenge: insights and quantitative and qualitative results. In: IEEE Intelligent Systems, pp. 1541–1672 (2016)
11. Walcott-Bryant, A., Kaess, M., Johannsson, H., Leonard, J.J.: Dynamic pose graph SLAM: long-term mapping in low dynamic environments. In: Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura, Portugal, pp. 1871–1878 (2012)
12. Santos, J.M., Krajnik, T., Fentanes, J.P., Duckett, T.: Lifelong information-driven exploration to complete and refine 4D spatio-temporal maps. IEEE Robot. Autom. Lett. **PP**(99), 1–14 (2016)
13. Montemerlo, M., Thrun, S.: Large-scale robotic 3-D mapping of urban structures. In: Experimental Robotics IX. Springer Tracts in Advanced Robotics, vol. 21, pp 141–150 (2006)
14. Hornung, A., Wurm, K., Bennewitz, M., Stachniss, C., and Burgard, W.: OctoMap: an efficient probabilistic 3D mapping framework based on octrees. Auton. Robots (2013)
15. Martín, F., Matelln, Lera, F.: Multi-thread impact on the performance of Monte Carlo based algorithms for self-localization of robots using RGB-D sensors. In: Workshop on Physical Agents (2016)
16. Dellaert, F., Fox, D., Burgard, W., Thrun, S.: Monte Carlo Localization for Mobile Robots. In: IEEE International Conference on Robotics and Automation (ICRA99), May 1999
17. Fox, D.: KLD-Sampling: adaptive particle filters and mobile robot localization. In: Advances in Neural Information Processing Systems (NIPS) (2001)